Planning in Description Logics: Deduction versus Satisfiability Testing

Liviu Badea
AI Research Lab
Research Institute for Informatics
8-10 Averescu Blvd., Bucharest, Romania
e-mail: badea@ici.ro

Abstract.
Description Logics (DLs) are formalisms for taxonomic reasoning about structured knowledge. Adding the transitive closure of roles to DLs also enables them to represent and reason about actions and plans. The present paper explores several essentially different encodings of planning in Description Logics. We argue that DLs represent an ideal framework for analysing and comparing these approaches. Thus, we have identified two essentially different *deductive* encodings (a "causal" and a "symmetric" one), as well as a *satisfiability*-based approach.

While the causal encoding is more appropriate for reasoning about precondition-triggered causal events, the symmetric encoding is more amenable to reasoning about possible outcomes of courses of actions without actually executing them (while allowing both progression and regression).

In the deductive approaches, the existence of a plan corresponds to an inconsistency proof rather than to a model of some formula. Viewing planning as satisfiability testing addresses this problem by reducing planning to model construction.

1 Introduction

Description Logics (DLs) are formalisms for taxonomic reasoning about structured knowledge. Like their predecessors (semantic networks and frame languages), DLs have been used mainly for representing and reasoning about the domain knowledge of a given problem, usually in the framework of a hybrid architecture.

Description Logics with the transitive closure of roles [2, 12] have also been proposed as a unifying formalism for various class-based representation languages as well as for representing tense, epistemic operators, actions and plans [7, 1, 3].

Some of these approaches rely on Schild's correspondence [12] between expressive description logics with the transitive closure of roles and propositional dynamic logic (PDL). Given that PDL has been conceived as a formal approach to reasoning about actions and dynamically evolving systems (such as programs), it may be surprising that so little research has been carried out towards representing planning in description logics.[1]

However, representing and reasoning about actions and planning in DLs is very important for modeling dynamically evolving DL knowledge bases at the conceptual level (as opposed to using an ordinary DL in a hybrid architecture, where one is not able to reason about actions *in* the DL, which is therefore incomplete).

The main goal of this paper is to present an in-depth analysis of the various approaches to encoding actions and planning in Description Logics. This issue is not entirely straightforward, since – contrary to a first impression – there are several essentially different ways of encoding actions and planning problems in DLs. For example, we can encode planning either as deduction or as satisfiability testing. Viewed as a deduction problem, we have identified two essentially different encodings of planning: a *"causal"* and a *"symmetrical"* one. These deductive approaches could also be used together in a realistic setting in which causal external events (even non-deterministic ones) as well as actions under the control of intelligent agents coexist.

2 The $\mathcal{ALC}^\star$ Description Logic

In the following, we consider the smallest description logic able to express actions and conditional plans, namely the regular closure $\mathcal{ALC}^\star$ of Schmidt-Schauß and Smolka's $\mathcal{ALC}$ language [13] extended with identities $id(C)$. Compared with other description logics, $\mathcal{ALC}^\star$ is quite expressive, since it allows the internalization of general (possibly cyclic) concept definitions by means of the transitive closure of roles.

The following concept and role constructors are available in $\mathcal{ALC}^\star$:

$$C ::= CN \mid \top \mid \bot \mid C_1 \wedge C_2 \mid C_1 \vee C_2 \mid \neg C \mid \langle R \rangle C \mid [R]C$$

$$R ::= RN \mid id(C) \mid R^- \mid R_1 \vee R_2 \mid R_1 \circ R_2 \mid R^*$$

where CN, RN are concept and role names respectively, $\langle R \rangle C$ are existential restrictions (usually written as $\exists R.C$), while

[1] We are considering description logics rather than plain PDL for encoding actions for two important reasons. First, description logics may provide additional constructs useful for integrating a theory of action in a more extensive KR framework. Second, in DLs it is possible to impose constraints on specific state instances (using assertional axioms). This is not possible in PDL.

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

$[R]C$ are value restrictions (written also as $\forall R.C$). Role union ($R_1 \vee R_2$), composition ($R_1 \circ R_2$) and reflexive-transitive closure (R^*) allow for regular role expressions, whereas the identity role construct $id(C)$ is useful for representing conditional plans. Role inverses (R^-) are needed for goal regression.

Recall that the transitive closure of roles is not expressible in first-order logic (it requires at least fixpoint logics). However it is essential not only for encoding general terminological axioms, but also for our encodings of planning in $\mathcal{ALC}^\star$.

In order to represent the symmetric encoding, we will need a more expressive DL, namely one that provides explicit fixpoint constructors. The $\mathcal{ALC}^\mu$ language [11, 5] is strictly more expressive than $\mathcal{ALC}^\star$ and provides the following additional concept constructors:

$$C ::= \mu X.C \mid \nu X.C \mid X$$

where X is a "fixpoint variable" which can occur only in the scope of the least/greatest fixpoint constructors $\mu X.C$ and $\nu X.C$ respectively. And although $\mathcal{ALC}^\mu$ admits no role constructors (besides role inverses), the $\mathcal{ALC}^\star$ role constructors (occurring in existential or value restrictions) can be expressed by means of fixpoints.

The terminological knowledge base consists of general concept implications of the form $C_1 \rightarrow C_2$, as well as validity axioms C (expressing the validity of the concept term C).

The assertional knowledge base consists of assertional axioms of the form

$s : C$ (concept instance assertions)
$(s, s') : R$ (role tuple assertions).

An interpretation satisfying the terminological and assertional axioms of a knowledge base (KB) is called a *model* of the KB. A KB is called *consistent* iff it admits a model and inconsistent otherwise. A concept C is called *satisfiable* w.r.t. a given KB iff it admits a non-void extension $C^\mathcal{I}$ in a model $\mathcal{I}$ of the KB. C is *valid* in a KB whenever $C^\mathcal{I} = \top^\mathcal{I}$ in all models $\mathcal{I}$ of the KB. C is valid iff its negation $\neg C$ is unsatisfiable.

Testing satisfiability (and therefore also validity) in $\mathcal{ALC}^\star$ as well as $\mathcal{ALC}^\mu$ is decidable, more precisely EXPTIME-complete [8, 5].

3 Encoding actions and planning in Description Logics

As we have mentioned in the introduction, Description Logics with the transitive closure of roles like $\mathcal{ALC}^\star$ can be used not only for representing taxonomic domain knowledge, but also for describing actions and plans. This can be achieved by regarding a DL role A as an action which transforms states S from (the extension of) the role's domain into states S' from (the extension of) its range: $(S, S') \in A^\mathcal{I}$. Thus, the value restriction $[A]C$ can be interpreted as the necessary precondition for action A to achieve the effect C.

Conditions (fluents) from our theory of action will be represented in a DL by concepts, while actions will be encoded as role names. Of course, (possibly conditional) plans can be represented as complex role terms, the role constructors $\vee, \circ$ and $*$ being interpreted as control structures (nondeterministic choice, sequence and nondeterministic iteration respectively). The identity role constructor $id(C)$ can be interpreted as a "test", which can be used for expressing the usual structured control primitives *if*, *while* and *repeat*.

In the following, we will deal with propositional STRIPS actions A described in terms of the following three condition sets (containing only non-negated fluents):

- preconditions Pre(A) (the conditions necessary for executing A)
- positive effects Add(A) (the fluents added by A's execution)
- negative effects Del(A) (the fluents deleted/falsified by A's execution).

The following relationships between the above condition-sets are assumed: $Pre(A) \cap Add(A) = \emptyset$ and $Del(A) \subseteq Pre(A)$.

For example, the simple blocks-world action `A = move_X_Y_Z` (which moves the block `X` from `Y` onto `Z`) admits the following STRIPS description: `Pre(A) = {on_X_Y, clear_X, clear_Z}`, `Add(A) = {on_X_Z, clear_Y}`, `Del(A) = {on_X_Y, clear_Z}`.

As we have already mentioned, there are several alternative approaches to encoding and reasoning about actions and plans in $\mathcal{ALC}^\star$. The two main categories of approaches are the *deductive* and the *satisfiability-based* ones. We start by discussing the deductive approaches.

3.1 Deductive planning in Description Logics

We have identified two essentially different encodings of planning as deduction: a *causal* (asymmetrical) one and a *symmetrical* one.

3.1.1 The causal (asymmetric) encoding

The causal encoding amounts to enforcing the existence of an action execution A whenever the preconditions $Pre(A)$ of A are verified:

$$[\mathsf{Eff}_{DED-CAUS}] \quad Pre(A) \rightarrow \langle A \rangle Add(A)$$

(where condition sets appearing in logical formulae are interpreted conjunctively).

The semantical interpretation of the above axiom[2]

$$holds(Pre(A), S) \rightarrow \exists S'.do(A, S, S') \wedge holds(Add(A), S')$$

shows that all actions A executable in state S (whose preconditions are satisfied in S) are actually executed in S, leading to (separate) successor states S'. The causal approach therefore encodes the entire search space (with all possible action executions from a given state) in its models.

Besides the explicit effects of action A, described by axiom $[\mathsf{Eff}_{DED-CAUS}]$, it is necessary to describe the persistence of the conditions (fluents) left unmodified by A. This is achieved by means of frame axioms of the form[3]

$$[\mathsf{Fr}_{DED}] \quad C \rightarrow [A]C \quad \textit{for all} \quad C \in Conditions - (Del(A) \cup Add(A)).$$

Note that since we are in a deductive setting it is not necessary to explicitly mention the deleted effects in the consequent of the above axiom. In other words, a stronger version like $Pre(A) \rightarrow \langle A \rangle (Add(A) \wedge \neg Del(A))$ is not needed

[2] We write $holds(C, S)$ instead of $S \in C^\mathcal{I}$ and $do(A, S, S')$ instead of $(S, S') \in A^\mathcal{I}$ in order to emphasize the fact that the interpretations of DL formulae are essentially situation calculus formulae.

[3] Since a given action typically affects only a small number of conditions, we will have to write $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{C}|)$ such frame axioms. Their number can be reduced to $\mathcal{O}(|\mathcal{C}|)$ by grouping the actions $A, A', A'', \ldots$ that leave C unaffected: $C \rightarrow [A \vee A' \vee A'' \vee \ldots]C$.

as long as the frame axioms do not allow the persistence of deleted effects. Similarly, a stronger version like $Pre(A) \rightarrow \langle A\rangle\top \wedge [A]Add(A)$ is also unnecessary for deductive planning.

A *planning problem* is usually specified by providing a (possibly incomplete) initial state described by the concept $Initial$ (a conjunction of the concept names representing the conditions initially true) and a final (goal) state $Final$. For example, in the Sussman anomaly problem `Initial = on_c_a` $\wedge$ `on_a_table` $\wedge$ `on_b_table` $\wedge$ `clear_c` $\wedge$ `clear_b` and `Final = on_a_b` $\wedge$ `on_b_c`.

The most straight-forward approach to such a problem would be to reduce it to proving a theorem of the form $Initial \rightarrow \langle ?Plan\rangle Final$ involving a meta-variable $?Plan$. Unfortunately, most description logic theorem provers do not allow for role variables (especially those with powerful role constructors, like $\mathcal{ALC}^{\star}$), so the simple approach above is not directly feasible.

If we knew the role term representing the plan $Plan = A_{i_1} \circ A_{i_2} \circ \ldots \circ A_{i_n}$, then the validity of the formula

$$Initial \rightarrow \langle Plan\rangle Final \qquad (1)$$

is equivalent with the validity of the plan.

However, since we do not know $Plan$, we need to try proving (1) for all possible action sequences $Plan$. Unfortunately, this cannot be done effectively, since there are infinitely many such action sequences and therefore infinitely many theorems to try proving. Therefore, we will consider reducing the problem to proving a *single* formula containing a disjunction of all possible action sequences:

$$\begin{aligned} Initial \quad \rightarrow \quad & Final \vee \langle A_1\rangle Final \vee \langle A_2\rangle Final \vee \ldots \vee \\ & \langle A_1 \circ A_1\rangle Final \vee \langle A_1 \circ A_2\rangle Final \vee \ldots \end{aligned} \qquad (2)$$

Since a disjunction of existential restrictions can be rewritten as an existential restriction $\langle R_1\rangle q \vee \langle R_2\rangle q = \langle R_1 \vee R_2\rangle q$, we can reduce (1) to

$$[\mathsf{Plan}_{DED-CAUS}] \quad Initial \rightarrow \langle Any^*\rangle Final$$

where $Any = A_1 \vee A_2 \vee \ldots \vee A_k$ is the disjunction of all atomic actions occurring in the problem (the "repertory of actions") [3, 6]. Note that the role term Any^* plays the role of the meta-variable $?Plan$.

The relationship between (1) and (2) is subtle and requires some explanations. In general, a proof of (2) does not entail the existence of a proof of (1) for some $Plan$ (although the reverse is true) because (2) requires that for each state S verifying $Initial$ we find a sequence of actions $Plan$ such that $\langle Plan\rangle Final$ holds – but $Plan$ need not be the same for all such states S!

The most straight-forward solution to this problem (pursued for example in [6][4]) would be to require complete state specifications (that do not allow for essentially different states S) and to make sure that the axioms constrain the successor states to be also completely specified. This amounts roughly to combining the axioms from our deductive (causal and symmetric) *and* SAT-based approaches. The problem with this approach lies in the large number of axioms employed which may significantly slow down a theorem prover, especially because reasoning with complete state specifications may be at a too fine-grained level, i.e. very close to "blind search" in the much too big space of complete state descriptions.

What we would like to achieve is to be able to reason with incomplete state specifications (for example by propagating only "weakest preconditions" and/or "strongest effects" instead of complete state information).

As shown above, incomplete state specifications give rise to situations in which a proof of (2) may construct a different $Plan$ for each completion (state) S verifying the incomplete initial state specification $Initial$. This ensures the existence of such a plan $Plan_S$ for each state S, but a given $Plan_S$ may not be applicable in all states S' verifying the incomplete specification $Initial$. On the other hand, the planning problem amounts to finding a plan that is guaranteed to work no matter what state we are in.[5]

Thus it may seem that it is impossible to reduce planning to proving a DL formula, so as to take advantage of an existing DL theorem prover. Therefore, it may seem we need to use a syntactical plan generation approach (like in [10]) by writing a specialized planning algorithm on top of a Description Logic (or Dynamic Logic) theorem prover. However, writing such a specialized planning algorithm external to the DL is somewhat inappropriate in a KR formalism like Description Logics, where we would like to be able to impose various constraints on the plan.

Fortunately, we can avoid this by showing that, although (1) and (2) are not equivalent in the general case, we can nevertheless recover a "global" plan (i.e. a solution to (1)) from a proof of (2). In order to do this, we shall single out a state S whose plan $Plan_S$ constructed according to (2) is also applicable to all the other states S'. The state S with this property is the completion of the (incomplete) initial state specification $Initial$ (obtained by conjoining to $Initial$ a negated literal $\neg C$ for each condition C not specified in $Initial$).

Due to our assumption that the precondition lists of actions contain only positive literals,[6] the negated literals in state descriptions do not influence the executability of actions (in the deductive settings, negated conditions are *not* propagated by frame axioms). Therefore, the plan $Plan_S$ for the completed state S will be applicable in all other states as well and will be a "global" plan. In our setting, (1) and (2) are therefore equivalent and we can safely reduce the planning problem to finding a proof for (2).

The planning problem has thus been reduced to proving the $\mathcal{ALC}^{\star}$ theorem $[\mathsf{Plan}_{DED-CAUS}]$. But proving the validity of such a formula is usually reduced in DLs to proving the

[4] De Giacomo and Lenzerini do not explicitly state that the initial state should be completely specified. However, their approach of reducing planning to proving the validity of $Initial \rightarrow \langle Any^*\rangle Final$ fails in the case of incompletely specified initial states due to their allowing actions with negated preconditions. For example, consider $Initial = p$, $Final = q$ and an action a with $Pre(a) = \{\neg q\}$, $Add(a) = \{q\}$, $Del(a) = \{\neg q\}$, described by means of the following axioms: $\neg q \rightarrow \langle a\rangle\top$; $\langle a\rangle\top \rightarrow \neg q$; $[a]q$. $Initial$ is incompletely specified since the value of q is not mentioned. Therefore, two possibilities arise: either q is true in $Initial$ (case in which the empty plan $Plan' = id$ is the only solution), or $\neg q$ holds in $Initial$ (case in which $Plan'' = a$ is the only solution), so there exists no "global" plan. But the formula $Initial \rightarrow \langle Any^*\rangle Final$ (i.e. $p \rightarrow \langle a^*\rangle q$) is nevertheless provable using the above axioms, showing that the approach in [6] fails in this case.

[5] "Conditional" plans like $Plan_S$ may be interesting in their own right, but we do not explore this issue further.

[6] If an action had a negated literal $\neg C$ as a precondition, we could replace it by the precondition C' and define $C' = \neg C$ as an axiom in the DL.

inconsistency of its negation:

$$[\neg \mathsf{Plan}_{DED-CAUS}] \quad Initial \wedge [Any^*]\neg Final.$$

Drawing an analogy with the answer-set of a logic programming query, we should be able to modify a DL theorem prover so that it returns a "falsifying interpretation" $\mathcal{I}$ for each inconsistent query $[\neg \mathsf{Plan}_{DED-CAUS}]$. This interpretation would be constructed while trying to build a model of the formula $[\neg \mathsf{Plan}_{DED-CAUS}]$. Whenever a plan exists, the latter formula is inconsistent due to a clash involving the goal condition $Final$ and the plan can be reconstructed from the (inconsistent) interpretation $\mathcal{I}$ built so far.

Note that unlike many planning systems which do not have a sound and complete stopping criterion,[7] the above approach to planning provides a decidable, sound and complete planning algorithm. This is especially important for proving that no plan exists.

The above reduction of plan construction to an inconsistency proof may seem somehow counter-intuitive in DLs, since we might have expected that a plan would correspond to a *model* of some formula rather than to a proof that no such model exists. This viewpoint will be pursued in the satisfiability-based encoding presented below.

The causal encoding presented above is more appropriate for reasoning about precondition-triggered causal events of the environment (as opposed to actions under the full control of agents – which may or may not choose to execute them, even if the preconditions are satisfied). It is also able to represent non-deterministic causal events (events with multiple possible outcomes). But since causal events are not necessarily reversible, the causal encoding is asymmetrical in a certain sense, and it does not allow a straight-forward representation of goal regression (i.e. reasoning backward from the goals $Final$). Reasoning in the causal encoding is therefore limited to progression (forward reasoning from the initial state), which may be inefficient (but it is the only type of reasoning possible when dealing with such precondition-triggered causal events).

3.1.2 The symmetrical encoding

The symmetrical encoding deals with representing the reasoning about possible outcomes of courses of action without actually executing the actions. More precisely, we shall write axioms saying that whenever the preconditions $Pre(A)$ of action A are verified and A is executed, the positive effects of A must hold in the successor state:

$$[\mathsf{Eff}_{DED-SYM}] \quad Pre(A) \rightarrow [A]Add(A).$$

This can be seen more easily in the semantic interpretation:

$$holds(Pre(A), S) \wedge do(A, S, S') \rightarrow holds(Add(A), S').$$

Similarly with the causal setting, we do not need to explicitly mention the deleted effects $\neg Del(A)$ in the consequent of the above axiom (because we are in a deductive setting).

The frame axioms $[\mathsf{Fr}_{DED}]$ are identical to the ones used in the causal setting.

Finally, the validity of a plan $Plan = A_{i_1} \circ A_{i_2} \circ \ldots \circ A_{i_n}$ is equivalent to proving the theorem $Initial \rightarrow [Plan]Final$. However, since we do not know $Plan$, we need to prove a formula containing a disjunction of all possible action sequences[8]

$$\begin{aligned} Initial \quad \rightarrow \quad & Final \vee [A_1]Final \vee [A_2]Final \vee \ldots \vee \\ & [A_1 \circ A_1]Final \vee [A_1 \circ A_2]Final \vee \ldots \end{aligned} \quad (3)$$

But unfortunately, the disjunction of value restrictions cannot be rewritten as a single value restriction,[9] so we *cannot* reduce (3) to a formula like $Initial \rightarrow [Any^*]Final$ (which would be the analog of $[\mathsf{Plan}_{DED-CAUS}]$). In fact, formula (3) cannot be encoded in $\mathcal{ALC}^\star$ (or PDL) and not even in *repeat*-PDL. In order to represent (3), we need the full expressive power of the μ-calculus, i.e. $\mathcal{ALC}^\mu$ (which provides general fixpoint constructors):

$$[\mathsf{Plan}_{DED-SYM}] \quad Initial \rightarrow \mu X.(Final \vee [A_1]X \vee \ldots \vee [A_k]X).$$

The validity of $[\mathsf{Plan}_{DED-SYM}]$ is equivalent with the inconsistency of

$$[\neg \mathsf{Plan}_{DED-SYM}] \quad Initial \wedge \nu X.(\neg Final \wedge \langle A_1 \rangle X \wedge \ldots \wedge \langle A_k \rangle X).$$

Using a result of Niwinski (mentioned in [14]) saying that the formula $\nu X.(\langle A_1 \rangle X \wedge \langle A_2 \rangle X)$ is not expressible in *repeat*-PDL, we conclude that neither $[\neg \mathsf{Plan}_{DED-SYM}]$ nor $[\mathsf{Plan}_{DED-SYM}]$ can be expressed in $\mathcal{ALC}^\star$ (not even in its ω-regular extension). Strangely enough, the symmetric encoding requires more expressive power than does the causal encoding. However, reasoning in $\mathcal{ALC}^\mu$ is just as hard/easy as reasoning in $\mathcal{ALC}^\star$ (both are EXPTIME-complete).

Regression The above encoding of planning seems to be more appropriate for progression (i.e. reasoning forward from the initial state and looking for a sequence of actions leading to the goal state). The following results show however that the above axioms can be rewritten in an equivalent form that is more appropriate for regression (backward reasoning from the final state by recursively replacing goals with action subgoals until they are satisfied in the initial state). This shows the intrinsic precondition-effect *symmetry* of the approach.

Proposition *The following axioms are equivalent:*
(1) $p \rightarrow [a]q$ *(2)* $\langle a^- \rangle p \rightarrow q$ *and (3)* $\neg q \rightarrow [a^-]\neg p$.

The "regressive" forms of the effect and frame axioms are therefore:

$$\begin{array}{ll} [\mathsf{Eff}^-_{DED-SYM}] & \langle A^- \rangle Pre(A) \rightarrow Add(A) \\ \quad or\ equivalently & \neg Add(A) \rightarrow [A^-]\neg Pre(A) \\ [\mathsf{Fr}^-_{DED}] & \langle A^- \rangle C \rightarrow C \\ \quad or\ equivalently & \neg C \rightarrow [A^-]\neg C. \end{array}$$

3.2 Planning as testing satisfiability in $\mathcal{ALC}^\star$

Viewing planning as satisfiability testing amounts to regarding a plan as a model of some formula rather than as a proof that no such model exists (as in the deductive approaches). Planning is thus reduced to model construction, in the spirit of [9]. But unlike Kautz and Selman, who reduce linear-time planning to propositional satisfiability, our approach reduces planning to $\mathcal{ALC}^\star$ satisfiability. A model corresponds thus to a Kripke structure rather than just a propositional truth assignment (as in [9]). Since $\mathcal{ALC}^\star$ provides the transitive closure of roles, we do not need to use (like [9]) iterative deepening

[7] They usually set an ad-hoc bound on the length of the plan.

[8] Similarly with the case of the causal setting.

[9] Note that $[R_1]q \vee [R_2]q \neq [R_1 \vee R_2]q = [R_1]q \wedge [R_2]q$.

over fixed-length planning problems. We additionally ensure the completeness of the termination check (our algorithms always terminate and in case they do so without finding a plan, then it is guaranteed that no such plan exists).

The effect and frame axioms used in the deductive approaches are correct and complete w.r.t. deduction, but they are not strong enough to rule out anomalous models. For example, they admit models in which actions are executed despite the fact that their preconditions are not satisfied. Such models can be avoided by using axioms of the form

$$[\mathsf{Pre}_{SAT}] \qquad \langle A\rangle\top \rightarrow Pre(A)$$
$$\textit{or equivalently} \quad [A^-]Pre(A).$$

For precondition-triggered causal events, we impose the executability axioms:

$$[\mathsf{Exec}_{SAT}] \quad Pre(A) \rightarrow \langle A\rangle\top.$$

The following axiom rules out models in which actions are executed but their effects do not hold:

$$[\mathsf{Eff}_{SAT}] \quad [A]\mathit{Eff}(A)$$

where $\mathit{Eff}(A) = Add(A) \wedge \neg Del(A)$ are the effects of action A.[10] Note that in the deductive setting, only the positive effects $Add(A)$ had to be enforced in the successor states of A. Even if these states would have been *consistent* with $Del(A)$, this would not have been sufficient for executing some other action whose preconditions are in $Del(A)$. $Del(A)$ should have been valid in those states and not just consistent with them.

The effect axiom $[\mathsf{Eff}_{DED-SYM}]$ in the symmetric deductive setting is weaker than its SAT counterpart $[\mathsf{Eff}_{SAT}]$ since $[\mathsf{Eff}_{SAT}]$ explicitly enforces $\neg Del(A)$ in the successor states of A and since $[\mathsf{Eff}_{DED-SYM}]$ constrains the successor states of A only if the current states verifies the preconditions $Pre(A)$.

$[\mathsf{Eff}_{DED-SYM}]$ is too weak for the SAT setting. However, the intermediate version $Pre(A) \rightarrow [A]\mathit{Eff}(A)$ is equivalent with $[\mathsf{Eff}_{SAT}]$ when combined with $[\mathsf{Pre}_{SAT}]$.

The frame axioms need to enforce the persistence not only of the positive literals (as in the deductive setting)

$$[\mathsf{Fr\text{-}pos}_{SAT}] \quad C \rightarrow [A]C$$
$$\textit{for} \qquad C \in Conditions - (Del(A) \cup Add(A))$$

but also of the negative literals

$$[\mathsf{Fr\text{-}neg}_{SAT}] \quad \neg C \rightarrow [A]\neg C$$
$$\textit{for} \qquad C \in Conditions - (Pre(A) \cup Add(A)).$$

The crucial difference w.r.t. the deductive approach consists in reducing the planning problem to testing the satisfiability of the formula

$$[\mathsf{Plan}_{SAT}] \quad Initial \wedge \langle Any^*\rangle Final$$

(or, equivalently, of its regressive variant

$$[\mathsf{Plan}^-{}_{SAT}] \quad Final \wedge \langle (Any^-)^*\rangle Initial.)$$

Therefore, a plan will be recovered from a *model* of the above formula. This requires practically no modification to an existing $\mathcal{ALC}^*$ consistency testing algorithm since such algorithms work by constructing models. In our tests, we have used the $\mathcal{R}eg\mathcal{AL}$ system described in [4] for solving propositional STRIPS planning problems encoded as satisfiability testing.[11]

Note that the SAT-based approach requires a completely specified initial state, in which either C or $\neg C$ holds for each condition C.[12] If neither C nor $\neg C$ holds in state S, then there may exist anomalous models in which actions having C as a precondition are executed in S. Fortunately, a completely specified initial state entails completely specified intermediate states.

4 Related work

Dynamic logic has been used in the past to encode reasoning about actions and plans [10], but a syntactical planning algorithm implemented on top of a Dynamic Logic theorem prover was usually employed. In the present paper we reduce planning to reasoning *within* a Description Logic, by using exclusively the DL reasoning services (without any additional external algorithms).

On the other hand [6] use an enhanced PDL for reasoning about concurrent actions. Their approach is very closely related to our asymmetrical deductive approach. However they use unnecessarily strong axioms to encode actions. As we have shown, a much weaker form of axioms is sufficient for planning in this setting. Also, reasoning with complete state information as in [6] may be too fine-grained, possibly affecting the efficiency of the approach. We are trying to propagate just enough informations in order to solve the planning problem.

REFERENCES

[1] Artale A., Franconi E. *A computational account for a description logic of time and action.* Proc. KR-94, 3-14.

[2] Baader F. *Augmenting concept languages by the transitive closure: An alternative to terminological cycles.* IJCAI-91, pp. 446-451, also DFKI RR-90-13.

[3] Badea Liviu. *A unified architecture for knowledge representation and reasoning based on terminological logics.* International Workshop on Description Logics, Roma 1995.

[4] Badea Liviu. *A unified architecture for knowledge representation based on description logics.* Proc. ECAI-96, 288-292.

[5] de Giacomo G., Lenzerini M. *Concept Language with Number Restrictions and Fixpoints, and its Relationship with the Mu-calculus.* Proc. ECAI-94, 411-415.

[6] de Giacomo G., Lenzerini M. *Enhanced Propositional Dynamic Logic for Reasoning about Concurrent Actions.* Proc. AAAI Spring Symposium, 1995.

[7] Devanbu P.T., Litman D.J. *Plan-based terminological reasoning.* Proc. KR-91, 128-138.

[8] Fischer M.J., Ladner R.E. *Propositional Dynamic Logic of Regular Programs.* Journal of Computer and System Science 18, pp.194-211, 1979.

[9] Kautz H., Selman B. *Pushing the envelope: planning, propositional logic, and stochastic search.* Proc. AAAI-96.

[10] Rosenschein S.J. *Plan synthesis: a logical approach.* Proc. IJCAI-81.

[11] Schild Klaus. *Terminological Cycles and the Propositional μ-Calculus.* DFKI Research Report RR-93-18, 1993.

[12] Schild Klaus. *A correspondence theory for terminological logics: preliminary report.* IJCAI-91.

[13] Schmidt-Schauss M., Smolka G. *Attributive concept descriptions with complements.* AIJ 48 (1), pp. 1-26, 1991.

[14] Streett R.S. *Fixpoints and Program Looping: Reductions from the Propositional Mu-Calculus into Propositional Dynamic Logic of Looping.* LNCS 193, 359-372, Springer 1985.

[10] $\neg Del(A)$ represents the conjunction of the negated conditions from $Del(A)$.

[11] An automated translation tool from STRIPS specifications to $\mathcal{ALC}^*$ axioms has been implemented for this purpose. Then, the $\mathcal{ALC}^*$ reasoning services are used for constructing a plan, i.e. a model of some formula.

[12] An incomplete initial state can be completed by adding a negated literal $\neg C$ for each unspecified condition C. This works since the condition sets $Pre(A)$, $Add(A)$ and $Del(A)$ contain only positive literals.